line card reduces the primary line card resources required for backup. Since backup line card 16n is not executing primary processes, more resources are available for backup. Hence, backup line card 16n executes six backup processes corresponding to six primary processes executing on primary line cards. In addition, backup line card 16n is partially operational and is executing device driver processes 490 and storing device driver backup state 498, 500 and 502 corresponding to the device drivers on each of the primary elements and network connection data 504, 506 and 508 corresponding to the network connections established by each of the primary line cards.

Alternatively, each primary line card could execute more or less than two backup processes. Similarly, each primary line card could execute no backup processes and backup line card 16n could execute all backup processes. Many alternatives are possible and backup processes need not be spread evenly across all primary line cards or all primary line cards and the backup line card.

Referring to Fig. 34b, if primary line card 16b experiences a failure, device drivers 490 on backup line card 16n begins using the device driver state, for example, DDS 498, corresponding to the device drivers on primary line card 16b and the network connection data, for example, CD 506, corresponding to the connections established by primary line card 16b to continue transferring network data. Simultaneously, backup line card 16n starts substitute primary processes 510' and 512' corresponding to the primary processes 474 and 475 on failed primary line card 16b. Substitute primary processes 510' and 512' retrieve active state from backup processes 510 and 512 executing on primary line card 16a. In addition, the slave SRM on backup line card 16n informs backup processes 526 and 524 corresponding to primary processes 472 and 473 on failed primary line card 16b that they are now primary processes. The new primary applications then synchronize with the rest of the system such that new network connections may be established and old network connections torn down. That is, backup line card 16n begins operating as if it were primary line card 16b.

Multiple Backup Elements:

also stored in memory local to the NMS client (for example, in proxies or GUI tables) and used by the NMS server to quickly retrieve data requested by the NMS client. Each NMS client, therefore, maintains its user context of interest, eliminating the need for client-specific device context management by the NMS server.

Referring to Fig. 59, an NMS client 850a runs on a personal computer or workstation 984 and uses data in graphical user interface (GUI) tables 985 stored in local memory 986 to display a GUI to a user (e.g., network administrator, provisioner, customer) after the user has logged in. In one embodiment, the GUI is GUI 895 described above with reference to Figs. 4a-4z, 5a-5z, 6a-6p, 7a-7y, 8a-8e, 9a-9n, 10a-10i and 11a-11g. When GUI 895 is initially displayed (see Fig. 4a), only navigation tree 898 is displayed and under Device branch 898a a list 898b of IP addresses and/or domain name server (DNS) names may be displayed corresponding to network devices that may be managed by the user in accordance with the user's profile.

If the user selects one of the IP addresses (e.g., 192.168.9.202, Fig. 4f) in list 898b, then the client checks local memory 986 (Fig. 59) for proxies (described below) corresponding to the selected network device and if such proxies are not in local memory 986, the NMS client sends a network device access request including the IP address of the selected network device to an NMS server, for example, NMS server 851a. The NMS server may be executed on the same computer or workstation as the client or, more likely, on a separate computer 987. The NMS server checks local memory 987a for managed objects corresponding to the network device to be accessed and if the managed objects are not in local memory 987a, the NMS server sends database access commands to the configuration database 42 within the network device corresponding to the IP address sent by the NMS client. The database access commands retrieve only data corresponding to physical components of the network device.

In one embodiment, data is stored within configuration database 42 as a series of containers. Since the configuration database is a relational database, data is stored in tables and containment is accomplished using pointers from lower level tables (children)

object and sends it to the NMS client which updates the GUI tables to display the added virtual ATM interface (e.g., 947c, Fig. 5u) to Virtual ATM Interfaces tab 947. The configuration object may be temporarily stored in local memory 986. However, once the GUI tables are updated, the NMS client deletes the configured object from local memory 986.

Because there may be many upper layer network protocol interfaces in network device 540, the port managed object and port proxy may become very large as more and more function calls (e.g., Add Virtual ATM Interface, Add Virtual MPLS Interface, etc.) are added for each type of interface. To limit the size of the port managed object and port proxy, all interface function calls may be added to logical proxies corresponding to logical upper layer protocol nodes. For example, an ATM node table 999 (Fig. 60n) may be included in configuration database 42, and when ATM service is first configured by a user on network device 540, the NMS server assigns an ATM node LID 999a (e.g., 5000) and inserts the ATM node LID and the managed device PID 999b (e.g., 1) in one row 999c in the ATM node table. The NMS server may also insert any attributes (A1-An). The NMS server then retrieves the data in the row and creates an ATM logical managed object (ATM LMO). Like the physical managed objects, the ATM logical managed object includes the assigned LID (e.g., 5000), attribute data and function calls. The function calls include Get Proxy and interface related function calls like Add Virtual ATM Interface. The NMS server stores the ATM LMO in local memory 987a and issues a Get Proxy function call. After creating the ATM proxy (ATM PX), the NMS server sends the ATM proxy to memory 986 local to NMS client 850a. The NMS client uses the ATM proxy to update GUI tables 985, and then uses it to later make function calls to get ATM interface related data from configuration database 42.

Thus, after the user selects OK button 950e (Fig. 5t) in virtual ATM interfaces dialog box 950, the NMS client places an "Add Virtual ATM Interface" function call to the ATM node proxy. The function call includes the ATM interface LID (stored in the GUI table), the corresponding ATM node LID and the parameters provided by the user through the ATM interfaces dialog box. The function call causes the NMS client to send JAVA RMI

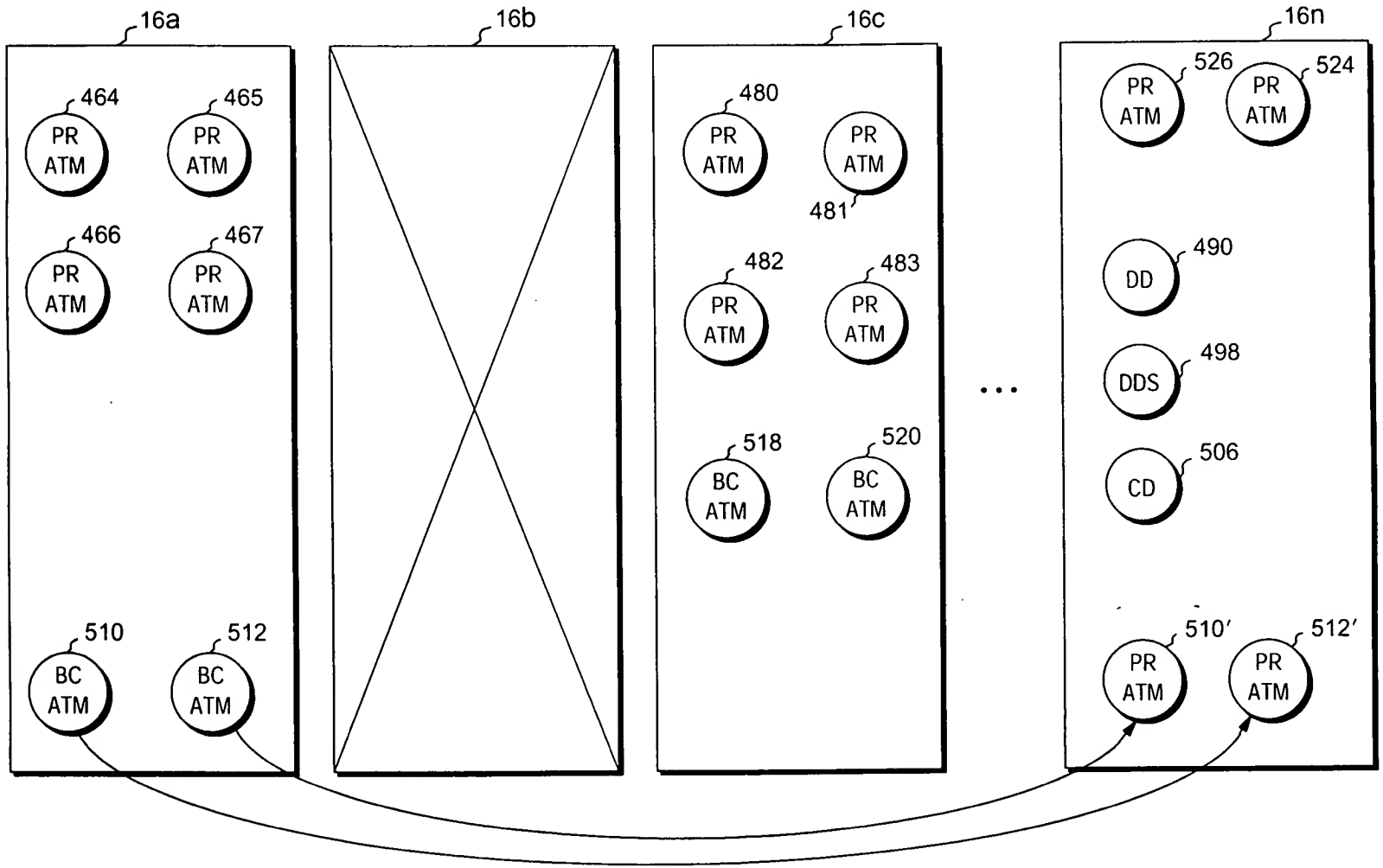
16a
16b
16c
16n
464
PR ATM
465
PR ATM
466
PR ATM
467
PR ATM
510
BC ATM
512
BC ATM
480
PR ATM
481
PR ATM
482
PR ATM
483
PR ATM
518
BC ATM
520
BC ATM
526
PR ATM
524
PR ATM
490
DD
498
DDS
506
CD
510'
PR ATM
512'
PR ATM

FIG. 34B

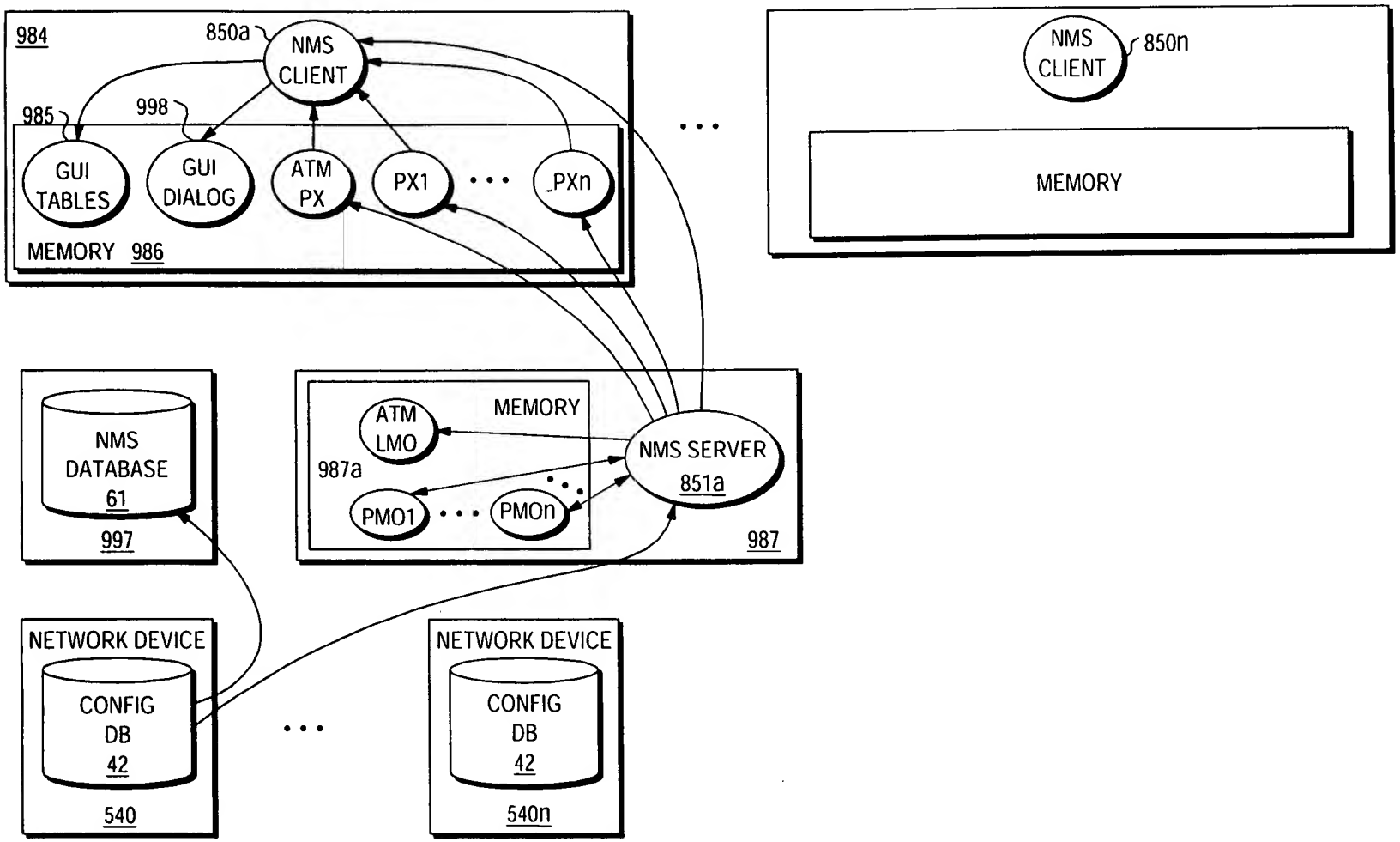
984
850a
NMS CLIENT
985
998
GUI TABLES
GUI DIALOG
ATM PX
PX1
PXn
MEMORY 986
NMS CLIENT
850n
MEMORY
NMS DATABASE 61
997
ATM LMO
MEMORY
987a
NMS SERVER 851a
PMO1
PMOn
987
NETWORK DEVICE
CONFIG DB 42
540
NETWORK DEVICE
CONFIG DB 42
540n

FIG. 59

VIRTUAL ATM IF TABLE 993

993a

993b

LID	A1	• • •	An	ATM IF LID
7489				5054
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

FIG 60J

VIRTUAL CONNECTION TABLE 994

994a

994b

LID	A1	• • •	An	VIR. ATM IF LID
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

FIG 60K

VIRTUAL LINK TABLE 995

995a

995b

995c

LID	A1	• • •	An	VIR. CONN. LID	CROSS. CONN. LID
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•

FIG 60L

CROSS-CONNECT TABLE 996

996a

996b

996c

LID	A1	• • •	An	VIR. LINK1 LID	VIR. LINK2 LID
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•

FIG 60M

ATM NODE TABLE 999

999a

999b

999c

LID	A1	• • •	An	MANAGED DEVICE PID
5000				1

FIG 60N